Flexible integration of path-planning capabilities

Iain C. Stobie, Milind Tambe and Paul S. Rosenbloom
USC/ Information Sciences Institute
4676 Admiralty Way
Marina del Rey, California 90292

November 1992
ISI/RR-93-348

REPORT DOCUMENTATION PAGE

FORM APPROVED
OMB NO. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE November 1992	3. REPORT TYPE AND DATES COVERED Research Report

4. TITLE AND SUBTITLE Flexible integration of path-planning capabilities	5. FUNDING NUMBERS N00014-91-J-1624 and DABT63-91-C-0025
6. AUTHOR(S) Iain C. Stobie, Milind Tambe and Paul S. Rosenbloom	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) USC INFORMATION SCIENCES INSTITUTE 4676 ADMIRALTY WAY MARINA DEL REY, CA 90292-6695	8. PERFORMING ORGANIZATON REPORT NUMBER RR-348
9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES) ARPA 3701 Fairfax Drive Arlington, VA 22203	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
12A. DISTRIBUTION/AVAILABILITY STATEMENT UNCLASSIFIED/UNLIMITED	12B. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Robots pursuing complex goals must plan paths according to several criteria of quality, including shortness, safety, speed and planning time. Many sources and kinds of knowledge, such as maps, procedures and perception, may be available or required. Both the quality criteria and sources of knowledge may vary widely over time, and in general they will interact. One approach to address this problem is to express all criteria and goals numerically in a single weighted graph, and then to search this graph to determine a path. Since this is problematic with symbolic or uncertain data and interacting criteria, we propose that what is needed instead is an integration of many kinds of planning capabilities. We describe a hybrid approach to integration, based on experiments with building simulated mobile robots using Soar, an integrated problem-solving and learning system. For flexibility, we have implemented a combination of internal planning, reactive capabilities and specialized tools. We illustrate how these components can complement each other's limitations and produce plans which integrate geometric and task knowledge.

14. SUBJECT TERMS Artificial Intelligence, Planning, Path Planning, Soar, Multiple criteria, Integration	15. NUMBER OF PAGES 10
	16. PRICE CODE

17. SECURITY CLASSIFICTION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UNLIMITED

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reoprts. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to stay within the lines to meet optical scanning requirements.

Block 1. Agency Use Only (Leave blank).

Block 2. Report Date. Full publication date including day, month,a nd year, if available (e.g. 1 jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element numbers(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	PR	- Project
G	- Grant	TA	- Task
PE	- Program Element	WU	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the repor.

Block 9. Sponsoring/Monitoring Agency Names(s) and Address(es). Self-explanatory

Block 10. Sponsoring/Monitoring Agency Report Number. (If known)

Block 11. Supplementary Notes. Enter information not included elsewhere such as: *Prepared in cooperation with...; Trans. of ...; To be published in...* When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD	- See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE	- See authorities.
NASA	- See Handbook NHB 2200.2.
NTIS	- Leave blank.

Block 12b. Distribution Code.

DOD	- Leave blank.
DOE	- Enter DOE distribution categories *from the Standard Distribution for* Unclassified Scientific and Technical Reports.
NASA	- Leave blank.
NTIS	- Leave blank.

Block 13. Abstract. Include a brief (Maximum 200 words) factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code (NTIS only).

Blocks 17.-19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contins classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

Table of Contents

In MOBILE ROBOTS VII: November, 1992

Flexible integration of path-planning capabilities

Iain C. Stobie
Department of Computer Science, Unversity of Southern California
Los Angeles, CA 90089

Milind Tambe
School of Computer Science, Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, Pa 15213

Paul S. Rosenbloom
Department of Computer Science and Information Sciences Institute,
University of Southern California,
4676 Admiralty Way, Marina del Rey, CA 90292

Abstract

Robots pursuing complex goals must plan paths according to several criteria of quality, including shortness, safety, speed and planning time. Many sources and kinds of knowledge, such as maps, procedures and perception, may be available or required. Both the quality criteria and sources of knowledge may vary widely over time, and in general they will interact. One approach to address this problem is to express all criteria and goals numerically in a single weighted graph, and then to search this graph to determine a path. Since this is problematic with symbolic or uncertain data and interacting criteria, we propose that what is needed instead is an integration of many kinds of planning capabilities. We describe a hybrid approach to integration, based on experiments with building simulated mobile robots using Soar, an integrated problem-solving and learning system. For flexibility, we have implemented a combination of internal planning, reactive capabilities and specialized tools. We illustrate how these components can complement each other's limitations and produce plans which integrate geometric and task knowledge.

1. INTRODUCTION

An autonomous mobile agent must plan and follow paths which are good fits to the problem at hand. For example, a short path is desirable for saving time or energy, but may be too expensive to compute exactly, and in hostile environments may often be unsafe. Many of the characteristics of the task and situation place requirements on the planning capabilities, as summarized in Table 1. Humans respond to this complexity by planning their motions in a wide range of ways, using learned routines, dead-reckoning corrected by landmark navigation, verbal or written instructions, and diverse kinds of maps[1, 2]. Animals also exhibit many methods for navigation[3].

Task characteristic	Requirements on planning capabilities
Path quality criteria	Produce path which optimizes criteria
Time-critical situation	Plan path in real time
Uncertain knowledge	Detect plan failure and replan, or use methods robust under uncertainty
Kinds of knowledge	Use method appropriate for representation and content
Nature of environment	Use method appropriate for regularities, e.g., natural vs. man-made
Interaction with task planning	Produce path satisfying task constraints, e.g., path descriptions useful for task-planning
Nature of actions	Produce path satisfying action constraints, e.g., no sharp turns
Nature of sensing	Produce plan satisfying sensing constraints, e.g., include active sensing

Table 1: Task characteristics place requirements on planning capabilities.

Current autonomous robots perform only simple tasks, and so require few of these capabilities, with the result that only one path-planning method is usually sufficient. However, as the complexity of tasks performed by future robots increases — for example, as required of a multi-function (semi-)autonomous household or search-and-rescue robot — more planning capabilities will also be needed by them. Different kinds of paths will be needed at different times, depending on the goals and environment, and may have to be produced in real-time. Several kinds and sources of knowledge will be available to the path planner, which in turn might have to provide different kinds of path descriptions to the task reasoner. Our hypothesis is that these complexities indicate that one planning method will not be sufficient, and instead a coordinated set of planning capabilities should be flexibly integrated with the task reasoner.

We have been testing this hypothesis in a simulated testbed, on tasks with several dynamically changing criteria and available sources of knowledge. Although our work is preliminary, it is apparent that there are tradeoffs involved in different integration strategies. At one extreme is the use of black-box path planners; that is, opaque subroutines which compute trajectories between destinations specified by the task reasoner[4, 5, 6]. The trajectories are then passed unmodified to the action component. This approach works best in situations where (1) the knowledge required to plan a path is limited and separable from the task knowledge; (2) the planned path is only to be used for direct execution, and need not be understood (as for explanation or replanning purposes) by the task planner; (3) it is straightforward for the task planner to formulate the relevant planning criteria in a form usable by the black box; and (4) for every set of joint planning criteria, there is a black box that can handle the entire set. To the extent that (1) fails to hold, either the path planner will perform with an impoverished knowledge base or knowledge must be duplicated between the two "modules". To the extent that (2) fails to hold, the task planner must understand much about how the black box works in order to understand and make use of its output. To the extent that (3) fails to hold, the task planner is faced with a complex formulation task that may involve obscure translations of knowledge into forms understood by the black box. To the extent that (4) fails to hold, the task planner may have to partition the problem across multiple black boxes — or do more of the task itself — and then understand, adapt and merge the results returned by the various boxes.

At the other extreme is a single general path planner, fully integrated with the general task reasoner. In this case the planner is as appropriate as the robot's knowledge allows, the development of the plan should be understandable, and there should be no extra formulation, translation, adaptation, or merging costs. However, for highly regular but computationally intensive tasks — which includes many special cases of the general path-planning task — black boxes can be much faster than general methods.

In this paper, we concentrate on a hybrid approach that combines the use of black-box path planners for specialized subtasks with a general planner that handles both the overall task plus additional path-planning subtasks. Essentially the general planner can be viewed as a combination of a task planner plus a set of glass-box path-planning specialists that share the task planner's knowledge and control (and, as described later, learning abilities). The general planner is responsible for formulating path-planning problems; selecting specialists (of both the black-box and glass-box variety); and understanding, adapting, merging, and utilizing their results.

In the remainder of this paper we illustrate the hybrid approach via an example scenario (Section 2), describe a simulated agent that uses the hybrid approach to produce the behavior in the scenario (Section 3), describe its implementation within the Soar architecture (Section 4), return to the key issue of combining multiple path criteria (Section 5), and conclude (Section 6).

2. TRAVERSING WHILE EVADING

The overall task is a safe-traversal task with a two-agent pursuit-evasion component. The agent of interest — that is, the one to be controlled via a Soar-based hybrid (path-)planner — is an *evasion* agent that must travel between two points in a simulation environment, while evading a second agent (the *pursuit* agent) that is attempting to pursue and catch it. While still relatively simple, this task is rich enough to have significant spatial and non-spatial aspects (such as using a model of the pursuit agent); multiple sources of knowledge; and multiple, dynamically changing, path-planning criteria. These, and other, properties should become clearer as we go through the example scenario.

The simulation environment provides a continuous real-time ground world with obstacles (walls) which block movement and vision, and hills which obstruct vision. A metric map for the environment is available. Potential sources of uncertainty in the environment include inaccuracies in the map and limitations on vision and actions. The pursuit agent, which can be considered part of the environment as far as the evasion agent is concerned, has a simple behavioral control program to chase the evasion agent if it is visible, and to wander otherwise.

A trace of an example scenario can be seen in Figure 1. The thick horizontal and vertical lines are the walls. The dotted line plots the movement of the pursuit agent, and the thin line plots the movement of the evasion agent. `EA start` and `PA start` indicate the start locations of the evasion and pursuit agent respectively. The star marked with `EA goal` indicates the goal location of the evasion agent. Initially the evasion agent plans a safe, short route to its goal: safe in that it attempts to avoid potential ambush places with poor escape possibilities. It thus avoids moving close to ends of walls, which can serve as ambush points. While executing the planned path, the evasion agent perceives the pursuit agent — the point labeled as `PA visible`. Simultaneously, the pursuit agent perceives the evasion agent. The evasion agent reacts immediately by starting to run away, while simultaneously deciding whether to hide or continue towards the goal. Here it chooses to hide, heuristically selecting a hiding place and generating a plan by which to get there. It then starts following the plan to the hiding place, with the pursuit agent attempting to chase it down. In this case, the evasion agent is successful in hiding behind the vertical wall shown, i.e., the pursuit agent is unable to see it. In general, the hiding strategies can be more complex than the one shown here — without the vertical wall, the evasion agent would have attempted to hide behind the horizontal wall in the top, right-hand corner.

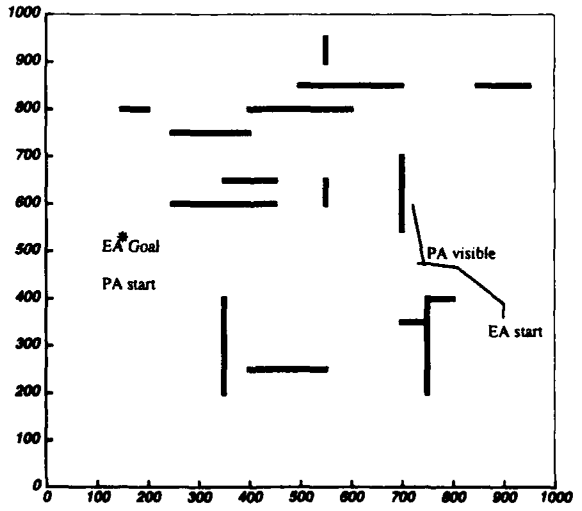

Figure 1: Example Scenario: EA is the evasion agent and PA is the pursuit agent.

Once at its hiding place, the evasion agent has several options as to what to do next. The first option is to continue to run away. This option is chosen if the pursuit agent is still visible, and thus likely to still be in pursuit. The second option is to simply race directly to the goal via the shortest path (ignoring safety and all other considerations). Then, if the evasion agent can get safely to its goal, it will. This option is chosen if it is estimated that the evasicn agent could get to its goal before being chased down by the pursuit agent (based on a simple model of the pursuit agent). The third option is to wait a while at the hiding spot, giving the pursuit agent a chance to wander off. This option is taken if the pursuit agent is known to be very close to the goal, disallowing the evasion agent from reaching it. The fourth option is to replan a safe, short path from the current position to the goal location. While replanning, the last seen position of the pursuit agent is taken into account — the pursuit agent takes time to wander away from its last seen position. This fourth option is the one taken here. Figure 2 shows the replanned path in the scenario. This replanned path is a longer detour to the goal, but it avoids the area around the pursuit agent, and allows the evasion agent to reach its goal safely. However, had the pursuit agent again intercepted the evasion agent, then the evasion agent would have repeated the cycle of hiding and replanning. Note that the evasion agent stores the result of all its planning activity in its memory. As a result, when re-confronted with similar later situations, the stored plans directly provide results, and thus avoid any planning effort.

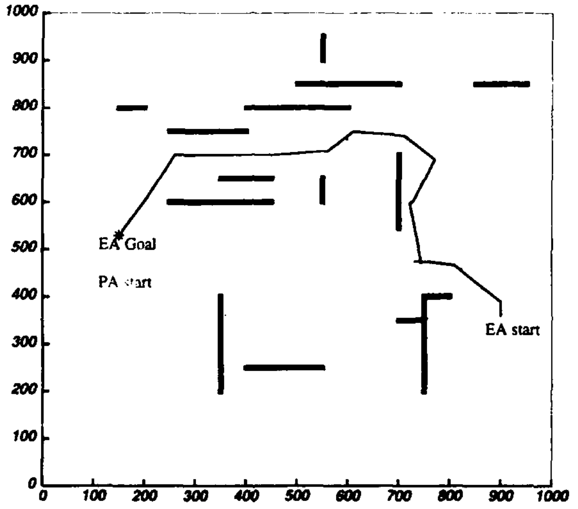

Figure 2: Replanning to reach the goal (scenario continued from Figure 1).

This scenario demonstrates how a number of the capabilities listed in Table 1 are required (and provided) by the evasion agent: planning paths for multiple quality criteria (plan length, obstacle avoidance, planning time, safety from (ambush by) a currently invisible adversary, and safety from an adversary in close pursuit); real-time planning (planning to hide, particularly in combination with a "running away" behavior); coping with uncertainty (about the other agent's location); use of several kinds of knowledge (a map, visual input, previous plans, heuristics, and an adversary model); and interaction with task planning. Some paths must meet several different criteria at the same time (such as path length and safety from a currently invisible adversary or path length and planning time), and the task requires paths with different sets of criteria at different times. The determination of safety involves analysis of possible ambush points and situation assessment to figure out what tactics the pursuit agent is using as well as counterplanning to offset those tactics (currently we assume a single tactic; addressing the issue of multiple tactics is an interesting issue for future work). Thus the evaluation function used by the path planners is a combination of spatial reasoning and agent modeling. Similarly, costs of actions vary with the situation; for example, hitting an obstacle when the pursuit agent is close is worse than hitting it otherwise.

Path planning in this scenario also uses different sources of knowledge in different ways. Planning a short, safe path can be done with just the map, but if vision or memory supplies the approximate location of the pursuit agent, that information together with expectations about the pursuit agent's behavior can be used to mark the pursuit agent's predicted locations as unsafe. Planning to hide uses both the map and the other agent's location, whereas the racing planner only uses the map.

3. THE HYBRID EVASION AGENT

The evasion agent's design is shown in Figure 3. It is constructed as a general task reasoner, plus a set of path-planning methods that are each specialized with respect to particular combinations of quality criteria. There are two glass-box specialists (a safety planner and a hiding planner), three black-box behaviors for (moving to an unobstructed destination, avoiding walls, and running away), and a black-box racing planner. The generalist invokes the specialists (both glass-box and black-box) by a switch, based on the situation. The specialists do not currently communicate directly with each other.

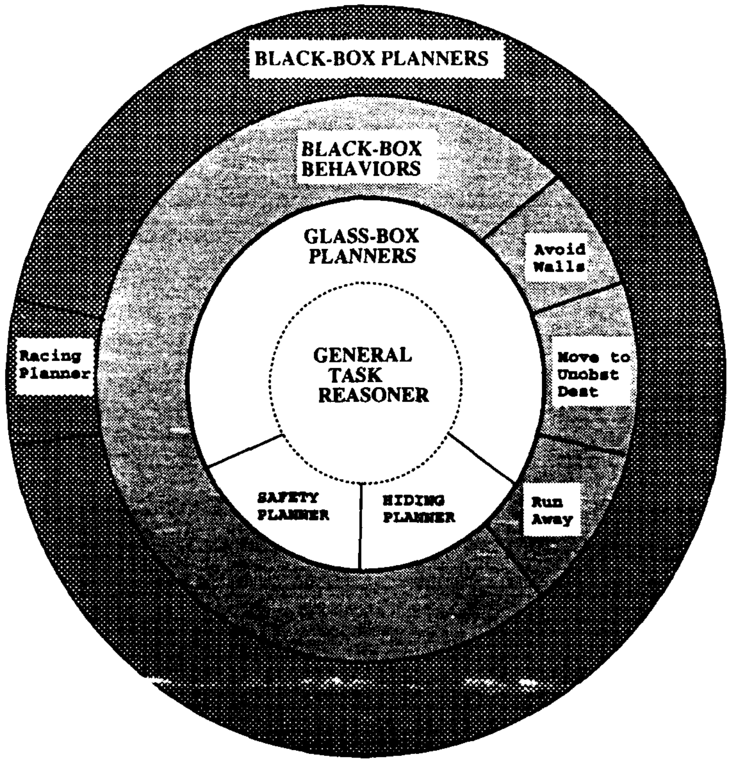

Figure 3: Integration of the general task-reasoner, the glass-box planners, and the black-box planners and behaviors (shaded regions).

The safety planner is invoked at the beginning of the task, and whenever replanning is chosen at a hiding place. It heuristically searches for a short, safe path from its current position to the goal position. The search is based on a map in which free space has been decomposed into regions between walls. Safety is then determined by the distance from possible ambush points — that is, from the ends of walls behind which the agent cannot see — and the space that is available for running away. If some approximate information about the pursuit agent is available, then that is used in determining safety as well. The search uses a greedy heuristic based on distance to the goal to order the safe regions at each choice point. In the process, the agent's learning mechanism stores away new rules that capture the results of searching the tree of paths, so the planner speeds up with experience. The final phase of the safety planner is to transform the generated sequence of regions into a sequence of midpoints of the edges of the regions. These edge midpoints form the ultimate plan to be followed.

The hiding planner is invoked while the evasion agent is running away from the pursuit agent (i.e., the "running away" behavior is active). It uses a set of heuristics based on the relative location of the pursuit agent and the nearby walls to generate potential hiding places. It then plans paths to the potential hiding points, but using a very different strategy than is used by the safety path planner. The safety path-planner is concerned with safety from a currently invisible adversary, about whom only some approximate information (if any) is available. In contrast, the hiding planner is concerned with safety from an adversary that is in visible range, and in close pursuit. Thus, for hiding it is important that the planning proceed quickly, and that the plan get the agent to its hiding place quickly. To generate a plan quickly, the hiding planner relies on pre-computed abstract plans which are instantiated for all the potential hiding positions. The instantiated plans are pruned and ranked, and the best one is executed. The abstract plans themselves are geared towards allowing the evasion agent to hide quickly, so as to allow it to move to its destination quickly. One aspect of this is that the abstract plans have success and failure conditions associated with them that help determine when they should be terminated early. If the plan succeeds prematurely, it can simply be terminated. If the plan fails prematurely, then it is abandoned and a new one is instantiated and executed. The other aspect of this is that, in contrast to safety plans which suggest moves between region-edge midpoints — to avoid obstacles and ambush points — hiding plans suggest moves between region-edge points that are close to walls, in order to minimize travel time. Thus, the hiding and safety planners use different strategies to convert regions into sequence of points, according to the relevant criteria — safety and shortness. Other approaches to converting a sequence of regions into a sequence of points are possible as well, e.g., making a heuristic adjustment to reduce jaggedness and other costs[7], or using a guaranteed minimal-length path planner[8].

The racing planner is invoked while the evasion agent is hidden, in order to determine the shortest path from the hiding position to th goal. It is implemented by the visibility-graph algorithm[9], using a map of grown obstacles (the robot is assumed to be circular). The racing planner is invoked if the goal and the pursuit agent are known to be in opposite directions, given the current position of the evasion agent, e.g., if the goal is to the north-east of the evasion agent, while the pursuit agent is to its south-west. The assumption here is that such a situation will allow the evasion agent to quickly reach the goal without worrying about safety. However, this heuristic may fail, making the path unsafe, e.g., in the above example, the only way to travel north-east may be first to travel south-west. We are currently exploring methods to address this possibility: the path output by the racing planner may need to be modified for safety; or the racing planner may be modified to provide multiple paths, and the unsafe ones rejected.

To maintain the uniformity of the system description, it is useful (and not terribly inappropriate) to think of behaviors — such as moving to an unobstructed destination, avoiding walls, and running away — as (degenerative) black-box path planners (and executors). The "moving to an unobstructed destination" behavior is invoked when a destination has been specified by one of the three high-level planners. It essentially plans the intermediate points on the path, while coping robustly with uncertainty along the way. The "avoiding walls" behavior is invoked when the evasion agent nears a wall. This rarely happens when following a safety plan, but can (and does) occur when following hiding and racing plans. The behavior is implemented as a variant of the potential fields approach, in which the walls exert repulsive forces[10]. The running away behavior is invoked when the pursuit agent is seen. It immediately causes the evasion agent to reverse direction if it is currently heading in the general direction of the pursuit agent. This buys time while hiding plans are being instantiated and selected.

Though, in general, the behavioral approach can be problematic due to the presence of local minima, in practice we have not found this to be a problem when used in conjunction with the other planners. We have found the behaviors to be particularly useful in coping with small-scale uncertainty and in allowing reasonable — though not deeply insightful — behavior to occur in parallel with planning.

4. IMPLEMENTING THE EVASION AGENT

The Soar architecture[11, 12] provides th· basis for the construction of the general planner and its associated glass-box planners, in addition to providing th· asic capabilities for planning, acting, reacting, and learning. It also provides the basis for the integration of the bl ..-box planners. Many different kinds of tasks have been programmed in Soar, including expert systems, natural language interpretation, video game players, robot control and visual attention[13, 14, 12, 15]. Tasks in Soar are organized into problem spaces; i.e., sets of operators which transform states. Problem solving progresses as a sequence of decisions that select problem spaces, states, and operators, and apply operators. These decisions are driven by a recogniti· n memory consisting of a parallel production system. If a decision cannot be made — because the immediate knowledge is inadequate — an impasse occurs and Soar creates a subgoal in which planning and other kinds of search can occur recursively. Soar automatically learns from these impasses by saving chunks that record a generalized form of the experience in working on a problem[16]. Input and output with the external world occurs asynchronously with decision-making, and passes through the same working memory as internal knowledge. Many standard problem solving methods have been shown to emerge naturally from programming task knowledge in Soar[17].

The generalist and its associated glass-box specialists are all realized as Soar problem spaces and thus fully share knowledge and control (and learning). If these specialists were implemented as black boxes, the possible improvement in speed would have to be traded off with costs of providing them input in the right format (particularly with respect to the pursuit agent) and translating the results produced by them. For example, the safety-from-attack criterion is currently computed dynamically using non-geometric knowledge (a model of the pursuit agent's likely behavior) and is combined with other criteria in a non-linear fashion. This evaluation changes depending on information about the pursuit agent. If this were implemented as a black-box, information about the pursuit agent and its likely behavior would have to be communicated to the black-box after translating the information into a format understood by the black-box. This communication would have to be done each time any new information about the pursuit agent became available.

Black-box methods and tools are integrated into Soar via I/O modules. The racing planner is constructed as a separate software system that Soar communicates with via a specially-constructed communication module. While this interaction has been quite simple so far, we are currently investigating more complex interactions with this planner. This topic — the use of external tools by intelligent systems such as Soar — has been analyzed by Newell and Steier[18], who defined six capabilities required: formulate-subtask, create-input, convert-output, interpret-result, operate-black-box, and simulate-black-box. The last one emphasizes that the task agent must have some expectation of what the black box does, just as people make ballpark estimates when using a calculator.

The black-box behaviors — moving to an unobstructed destination, avoiding walls, and running away — are all implemented directly as I/O modules (implemented in C), rather than as separate software systems.

5. COMBINING MULTIPLE PATH-CRITERIA

There are a large number of issues raised by the hybrid approach and its instantiation in the evasion agent, including how (and if) it can provide all of the task characteristics listed in Table 1. An in-depth discussion of all of these is beyond the scope of this paper. So we will have to be satisfied here with returning to just one of the key items: combining multiple quality criteria.

The most common approach to building path planners that can handle multiple quality criteria is to express the criteria numerically and combine them with a weighted linear sum. This has been used to combine distance and safety from obstacles[19, 7], and in the Weighted Region Planner (WRP) approach[20]. However, ıe underlying assumptions of independent criteria and context-free combination have many problems. First, some criteria are not easily expressed numerically (e.g., safety from an intelligent enemy) and doing so hides the task knowledge in the evaluation function, and is thus hard to explain and acquire. Often the numbers are just relatively crude approximations to the true symbolic criteria such as 'better', 'safe', 'short'. Second, even if the criteria are easily expressed numerically, it often makes no sense to add them since their actual combination may be a complex function of the context.

Multiple-criteria optimization has been extensively studied in Operations Research[21] and we suspect that more complex combination strategies will need to be implemented in path planning as the required tasks become more complex. One example of this from existing work in path planning is the Vector Field Histogram method of obstacle avoidance[22]. It uses the same concept of attractors and repulsors as the potential field method[10], but rejects the latter's weighted linear sum. Another existing example comes from work in terrain navigation: in Denton and Froeberg's system[23], each grid in a map is evaluated according to each criterion (such as mobility, concealment, and lethality), and these criteria are combined according to expert rules. An optimal route is then found by dynamic programming.

The approach we have taken in constructing the evasion agent is to organize the overall set of criteria used by the agent into multiple sets of conjunctive criteria. Each conjunctive set specifies a set of criteria that can be simultaneously active, and each such set is handled by a specialized path planner. For instance, the hiding planner handles the criteria of short plan-time, safety from a visible adversary in close pursuit, obstacle avoidance, and path length; while the safety planner handles safety from ambush by a currently invisible adversary, obstacle avoidance, and path length. Choice among the sets of criteria is handled by the task planner — it chooses criteria by choosing a specialized planner. Combination among the conjunctive sets is handled by the respective specialized planners. In particular, the safety and hiding planners use particular lexicographic orderings — the safety planner first filters out any path that violates either safety from ambush or obstacle avoidance, and then chooses among the remaining plans according to their length, and the hiding planner effectively filters out all plans that cannot be generated quickly and then picks among the remaining ones based on distance — while the avoid-walls behavior uses a simple summation of the potential fields generated by the various obstacles. The other planners are all single criterion planners.

As the tasks get more complex, and the need to respond more flexibly increases, we would ultimately expect this simple mapping of sets of criteria onto individual planners to break down. When this happens, the general planner will either need to handle the entire task itself, or it will need to be able to break down its sets of conjunctive criteria into subsets that can be handled by what specialized methods it does have, and then understand the results provided by the specialists so as to be able to combine them into a single solution reflecting the entire set (while perhaps handling some of the criteria itself).

Though it is not at all clear at this point how to do this in general, there are at least some approaches that have proven successful for particular special cases. Miller and Schubert[24], and Kambhampati et al.[25] have shown how a general-purpose planner can combine the results from multiple specialist planners, if the generalist is a least-commitment planner and the specialists always formulate their results as constraints. In Denton and Froeberg's system[23], the path found by dynamic programming is passed to a method which evaluates it according to certain global mission criteria and modifies it as needed. During execution, a third method then uses a local evaluation function to further modify the plan and change the robot's actions.

6. CONCLUSION

In this paper we have argued that an intelligent robot should integrate a number of different kinds of path-planning capabilities, and we have described and demonstrated one instance of such an integration. There is, however, still much to be done in producing a system that integrates together a set of planners capable of fully satisfying the requirements listed in Table 1. We focused in some detail here on the issues raised by one of these requirements — the combination of multiple path criteria — but at least as many issues are raised by others, such as the use of methods that are appropriate for a wide range of available knowledge. For example, more diverse sources of knowledge will require different planning methods — such as landmark navigation using instructions — and many more opportunities to use black boxes may be identified (such as using Voronoi diagrams to stay away from obstacles). It is also critical to better understand the trade-offs in using general planners, glass-box specialists and black boxes; and also to understand how a range of such planners should interact to provide effective performance in a range of complex tasks within richly structured, dynamic environments.

7. ACKNOWLEDGEMENTS

This work was sponsored by the Office of Naval Research (ONR) under contract number N00014-91-J-1624 and by the Defense Advanced Research Projects Agency (DARPA) under contract number DABT63-91-C-0025, modification number P0004. We would like to thank Charles Dolan of the Hughes AI Center for supplying the original simulation.

8. REFERENCES

1. Chase, W. G., "Spatial Representations of Taxi Drivers", in *Acquistion of Symbolic Skills,* Rogers, D. R., and Sloboda, A., ed., Plenum, New York, 1982.

2. Lynch, K., *The image of the city,* MIT Press, Cambridge, MA, 1960.

3. Gallistel, R., *The organization of learning,* MIT Press, Cambridge, MA, 1990.

4. Gat, E., "Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots", *Proceedings of the National Conference on Artificial Intelligence,* 1992.

5. Goodwin, R., and Simmons, R., "Rational handling of multiple goals for mobile robots", *Proceedings of the AAAI Planning Symposium,* 1992 pp. 70-72.

6. Mitchell, J. S. B., "Algorithmic approaches to optimal route planning", *Proceedings of the SPIE conference on Mobile Robots,* 1990.

7. Thorpe, C., "Path relaxation: path planning for a mobile robot", *Proceedings of the National Conference on Artificial Intelligence,* 1984, pp. 318-321.

8. Holmes, P. D., and Jungert, E. R. A., "Symbolic and geometric connectivity graph methods for route planning in digitized maps", *IEEE transactions on Pattern Analysis and Machine Intelligence,* Vol. 14, No. 5, 1992.

9. Lozano-Perez, T. and Wesley M. A., "An algorithm for planning collision-free paths among polyhedral obstacles", *Communications of the ACM,* Vol. 22, No. 10, 1979, pp. 560-570.

10. Khatib, O., "Real-time obstacle avoidance for manipulators and mobile robots", *International Journal of Robotics Research,* Vol. 5, No. 1, 1986, pp. 90-98.

11. Laird, J. E., Newell, A. and Rosenbloom, P. S., "Soar: An architecture for general intelligence", *Artificial Intelligence,* Vol. 33, No. 1, 1987, pp. 1-64.

12. Rosenbloom, P. S., Laird, J. E., Newell, A., and McCarl, R., "A preliminary analysis of the Soar architecture as a basis for general intelligence", *Artificial Intelligence,* Vol. 47, No. 1-3, 1991, pp. 289-325.

13. Laird, J.E. and Rosenbloom, P.S., "Integrating execution, planning, and learning in Soar for external environments", *Proceedings of the National Conference on Artificial Intelligence,* July 1990.

14. Newell, A., *Unified Theories of Cognition,* Harvard University Press, Cambridge, Massachusetts, 1990.

15. Wiesmeyer, M.D., *An Operator-Based Model of Human Covert Visual Attention,* PhD dissertation, University of Michigan, 1992, CSE-TR-123-92

16. Laird, J. E., Rosenbloom, P. S. and Newell, A., "Chunking in Soar: The anatomy of a general learning mechanism", *Machine Learning,* Vol. 1, No. 1, 1986, pp. 11-46.

17. Laird, J.E. and Newell, A., "A universal weak method: Summary of results", *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* August 1983, pp. 771-773.

18. Newell, A. and Steier, D., "Intelligent Control of External Software Systems", Tech. report EDRC 05-55-91, Engineering Design Research Center, Carnegie Mellon University, April 1991.

19. Suh, S., and Shin, K. G., "A variational dynamic programming approach to robot path-planning with a distance-safety criterion", *IEEE Journal of Robotics and Automation*, Vol. 4, No. 3, 1988.

20. Mitchell, J. S. B., "An algorithmic approach to some problems in terrain navigation", *Artificial Intelligence*, Vol. 37, 1988, pp. 171-201.

21. Hwang, C., and Masud, A. S. M., *Multiple objective Decision Making*, Springer Verlag, Berlin, 1979.

22. Koren, Y., and Borenstein, K., "Potential field methods and their inherent limitations for mobile robot navigation", *Proceedings of IEEE International Conference on Robotics and Automation*, 1991.

23. Denton, R. V., and Froeberg, P. L., "Applications of Artificial Intelligence in Automated Route Planning", *Proceedings of SPIE conference on applications of Artificial Intelligence*, 1984, pp. 126-132.

24. Miller, S. A., and Schubert, L. K., "Using specialists to accelerate general reasoning", *Proceedings of the National Conference on Artificial Intelligence*, 1988.

25. Kambhampati, S., Cutkosky, M., Tanenbaum, M., and Lee, S. H., "Combining specialized reasoners and general purpose planners: a case study", *Proceedings of the National Conference on Artificial Intelligence*, 1991.